Release Planning Document

COSC 4P02
Software Engineering II
Winter 2024

January 26th, 2024
Brock University

Team Structure

QiQi Gao Product Owner/Dev... qg17jh@brocku.ca 6416762	Michael Boulet Scrum Master/Dev... mb20ot@brocku.ca 7063548	Meet Patel Developer mp20dp@brocku.ca 7056708	Michael Noyes Developer mn17bg@brocku.ca 6261374
Hamza Yousuf Developer hs19uo@brocku.ca 6772149	Jonathan Coletti Developer jc21jq@brocku.ca 7333339	Jose Henriquez Developer jh20uo@brocku.ca 7088792	David Fawzy Developer df20ft@brocku.ca 7084593

Definitions

Herein, and in future documents, the following definitions apply:

- Issue tracker: product backlog
- Issue: backlog item, or user story
- Sprint goal: sprint backlog item

Project Synopsis

Creating a website and browser plug-in/extension that takes any URL and summarizes the content of the page. If the URL provided is a link to a youtube video, the website will summarize the video, through automated transcription and/or a combination of other more effective techniques. This summary is then output as a shortened link that, when entered as an address, redirects back to this summary.

Requirements

Functional Requirements (features)

- Allow users to register accounts on the app
- Allow users to authenticate themselves on the app
    - Via email,
    - and via Oauth
- Users (authenticated or not) must be able to:
    - Allow users to input URL of a webpage and receive a summary and a short link
    - Allow users to input URL of a video and receive a summary of the video, and a shortened link
    - Allow users to load a previously processed short link and view the summarized content
- Allow authenticated users to view a dashboard that displays a list of their previously processed content history,
- Paid users are able to select custom comprehension levels (e.g. summarize to the understanding on an undergraduate), select summary lengths (e.g. single sentence, paragraph, long essay),
- Let authenticated users install a web extension where they can quickly summarize pages
- Let paid authenticated users generate API keys and access API endpoints

Non-functional Requirements

- Less than 1% downtime
- Page loads should be within 10 seconds
- Summarization should accurately capture key information from the source content
- Interface should be user-friendly with instructions and feedback
- Should encrypt transactions with industry-standard encryption
- The server backend should be reasonably secure from intrusion
- SQL queries should be designed to avoid SQL injections

Domain Requirements

- Follow PIPEDA
- Supports content localization and multiple currencies

Software Engineering Process

Github will be our main code and scrum collaboration tool, responsible for hosting our code, issue tracker, pull requests, code review, and CI. Our team's public Github organization can be found here. Hosted therein is:

- the main project repository,
- the main project's issue tracker,
- the document/report tracking repository, and,
- the scrum tracking project.

Depending on the results of the requirements process, and what software architecture, language, and frameworks are selected as a result, the following may be implemented depending on feasibility and time constraints: code style guides, continuous integration, a production environment, automatic deployment, package managers, and additional architecture specific tools. These processes will be planned if and when their need arises.

Commiting Workflow

We will aim to have git branches follow a master/develop branches in combination with feature branches. The master branch will track the latest stable release, while develop tracks current development. Feature branches may branch off of develop and rebase back into develop when progress finishes. One-off changes may be directly committed to development. The master branch will not be directly committed to, and will be rebased from development when a new release is ready.

Feature branches and one-off commits can be reviewed by developers via Github's by creating a pull request. Other developers will then review it and either approve the pull request or request changes. A certain number of approvals can be set as required for a pull request to be merged.

Scrum

Our scrum sprint schedule will start on January 23rd, with each sprint lasting for 1 week, until either: the project concludes, or the term ends, whichever occurs first. Team sprint goals will be tracked on the respective github project on a per-sprint basis. Individual developer's goals for the sprint will be self-selected on every sprint planning session under the guidance of the product owner, and the current state of the issue tracker.

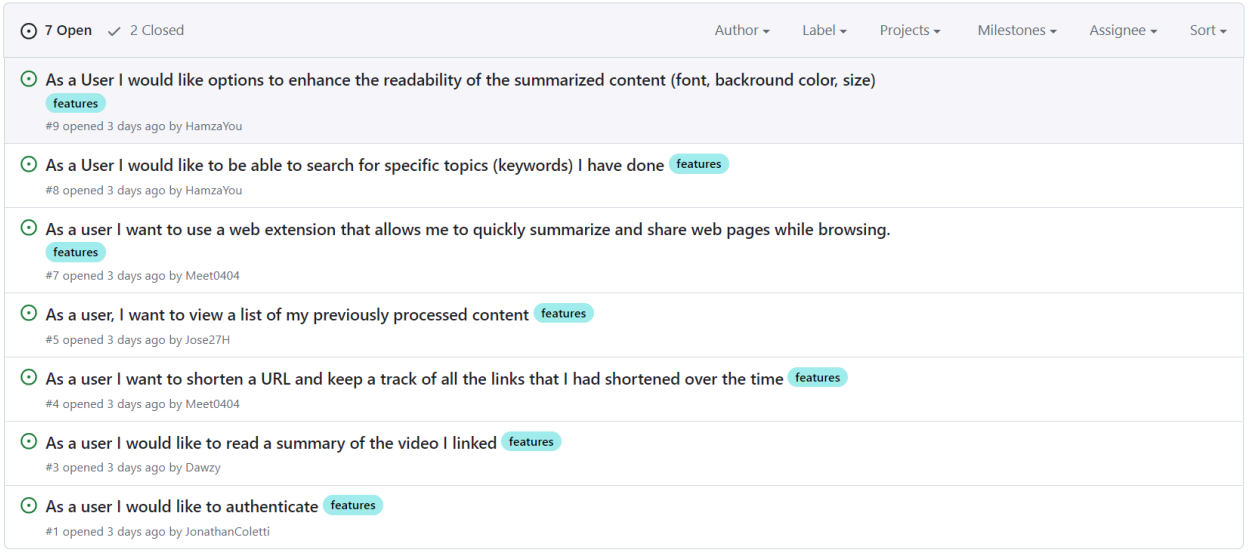
Recurring progress meetings should be held weekly every Tuesday at 20:30 UTC-05:00. On alternating progress meetings that coincide with a new sprint, the next sprint will be planned. Communication, including meetings, will be performed over Discord.

Issue Tracker

The issue tracker will be populated as the project progresses, with issues being added as they are identified. Issues may be tagged with the appropriate tags that identify & classify the issue. Features, other functionality, or tracking issues may be added to the issue tracker as they arise, with or without discussion. Tracking issues should either: identify issue numbers of their related issues, or preferably, enumerate all relevant issues under the tracking issue as subgoals. Finally, sprint goals may be converted to and linked to an issue using the appropriate github tool when appropriate.

After issues are identified on the tracker, on subsequent sprints, they may be worked on depending on their severity and relevance to the current sprint goals. Alternatively, if an issue arises that is found to block a current sprint goal's progress, work on the issue may be done immediately and should be noted in the sprint goal's description.

User Stories



Product Backlog

scrum

Table | Roadmap | Cards | + New view

sprint:@current

	Ass...	Title	Sp...	S...	Labels
	Dawzy 12				
1	Dawzy	Change default font	Sprint	D...	
2	Dawzy	Add dark mode	Sprint	T...	
3	Dawzy	Refactor <SearchBar /> into a reusable <Input /> ...	Sprint	T...	
4	Dawzy	Add primary & secondary button design	Sprint	T...	
5	Dawzy	History design	Sprint	T...	
6	Dawzy	Styling Page Guide	Sprint	T...	
7	Dawzy	Finalize page designs	Sprint	T...	
8	Dawzy	Finalize mobile view	Sprint	T...	
9	Dawzy	Validate URL before sending to server	Sprint	T...	
10	Dawzy	Validate login & sign up fields before sending to s...	Sprint	T...	
11	Dawzy	Implement auth	Sprint	T...	
12	Dawzy	Touch up with subtle animations using framer-mo...	Sprint	T...	
+	Add item				
	Dawzy, FieryAced, HamzaYou, JonathanColetti, Jose27H, mb20ot, Meet0404, and Mikeyy99 1				
13	Daw...	decide project name	Sprint	T...	
+	Add item				
	Dawzy and Meet0404 1				
14	Dawz...	Implement designed UI pages	Sprint	D...	
+	Add item				
	FieryAced 1				
15	Fiery...	I exist	Sprint	T...	
+	Add item				
	JonathanColetti 4				
16	Jonat...	setup packages	Sprint	T...	
17	Jonat...	Create fastAPI route points	Sprint	T...	
18	Jonat...	Create and implement the register endpoint	Sprint	T...	
19	Jonat...	Create the JWT auth method	Sprint	T...	
+	Add item				
	Jose27H and mb20ot 2				
20	Jose...	database implementation #15	Sprint	In...	features
21	Jose...	database table design #14	Sprint	In...	features
+	Add item				
	mb20ot 4				
22	mb2...	setup local environment	Sprint	In...	
23	mb2...	configure postgrest	Sprint	T...	
24	mb2...	design api	Sprint	T...	
25	mb2...	start issuing jwt tokens	Sprint	T...	
+	Add item				
	Meet0404 1				
26	Meet...	Hook up to backend endpoints	Sprint	T...	
+	Add item				
	Mikeyy99 1				
27	Mike...	create client and server validation	Sprint		

Contribution Tracking

Contributions will be mainly tracked by git commit history, and pull request/issue contributions. In the case that multiple authors worked on a commit, a co-author should be added to the commit. Attendance to meetings will be automatically tracked out-of-band by a Discord bot. Message history in discord channels will also be used as contribution tracking when deemed necessary.

Timeline

Sprint timeline:

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9	Sprint 10	Sprint 11	Sprint 12
Jan 23	Jan 30	Feb 6	Feb 13	Feb 20	Feb 27	Mar 5	Mar 12	Mar 19	Mar 26	Apr 02	Apr 09

Feature Timeline:

- Website prototype – sprint 1-2
    - Main pages
    - UI Functionality
    - Links direct to the right pages
- Database creation, design, functionality – sprint 1-3
    - Postgres setup
    - Table design
- Backend – Sprint 2-7
    - Authentication
- Link Shortening functionality – Sprint 3
- Scraping website/transcribing videos functionality – Sprint 3-6
- Summarization/LLM interfacing – Sprint 7
- Production – Sprint 8-11
    - Apache setup
    - Domains
    - CI
- Browser extension Sprint 10-12